Tabla de Contenido

1 Instalación

Después de descargar el plugin de la página del curso descomprima el archivo zip en eclipse/plugins, creando la carpeta co.edu.uniandes.cupi2.brazo_mecanico. Si Eclipse está en ejecución es necesario reiniciarlo.

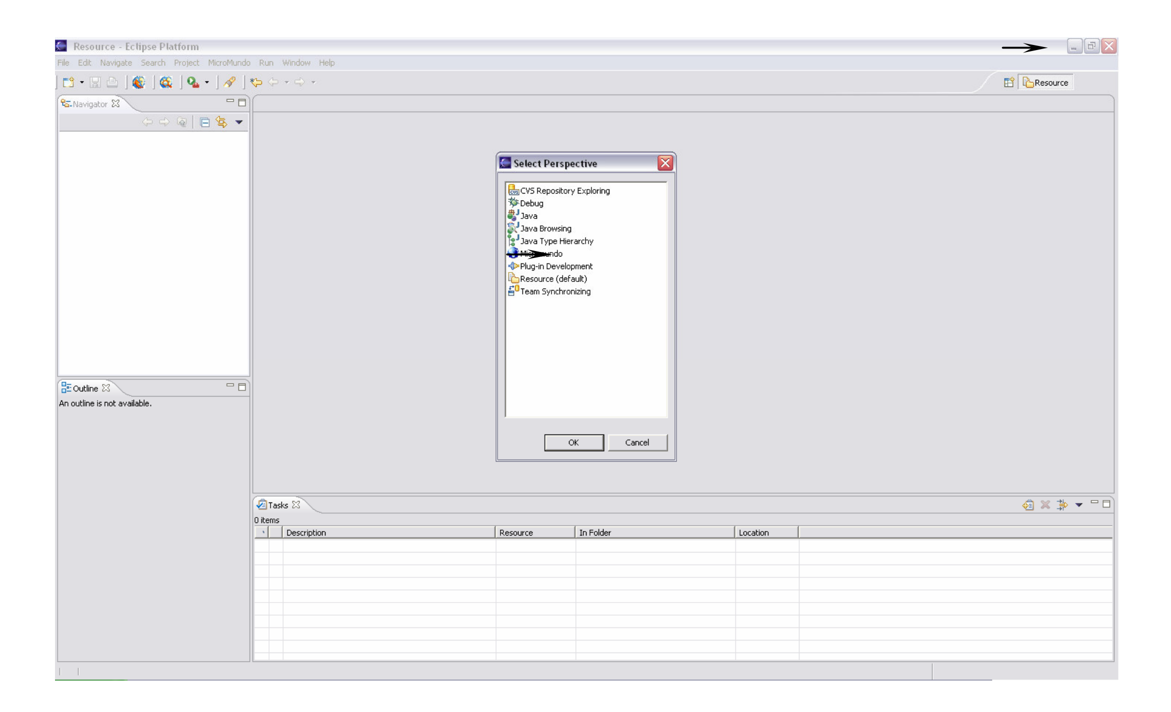
2 Introducción

El Entrenador Brazo Mecánico, es un Plugin para Eclipse el cual busca facilitar el aprendizaje de la algoritmia mediante el uso del un brazo mecánico, que interactúa dentro un mundo con cubos de distintos colores y valores.

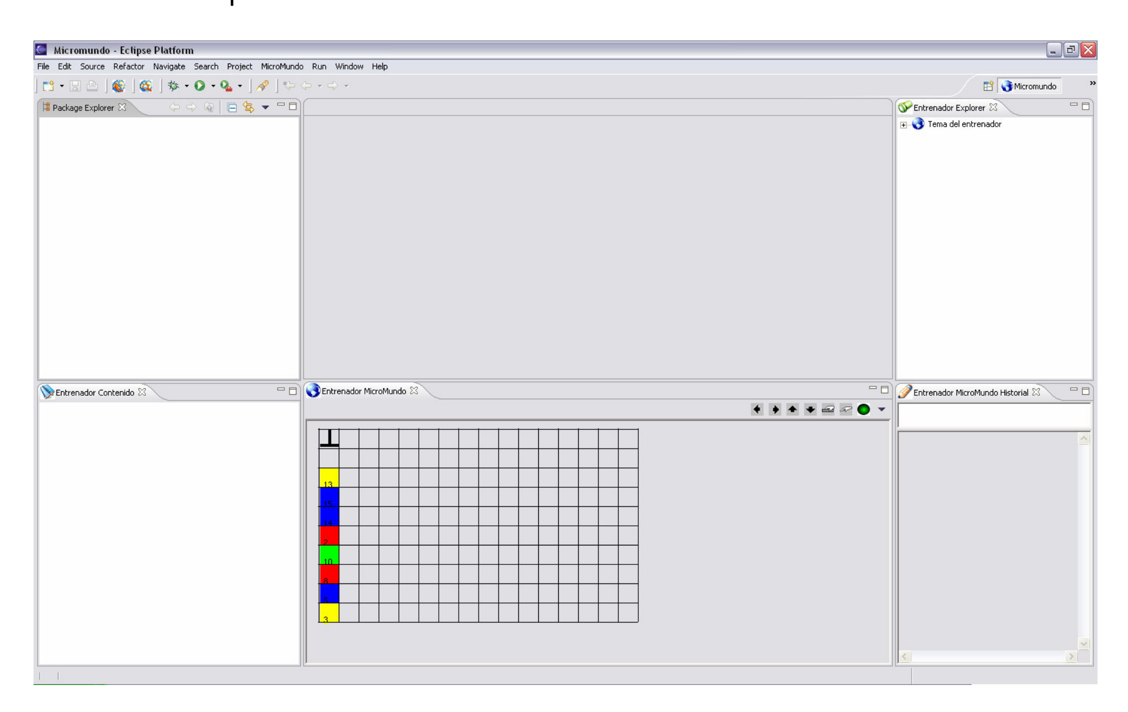
En Entrenador Brazo Mecánico se encuentra completamente integrado a Eclipse, este contiene una serie de vistas, contenidas dentro de una perspectiva de Eclipse, que permiten la visualización e interacción del Brazo Mecánico.

3 Perspectiva MicroMundo

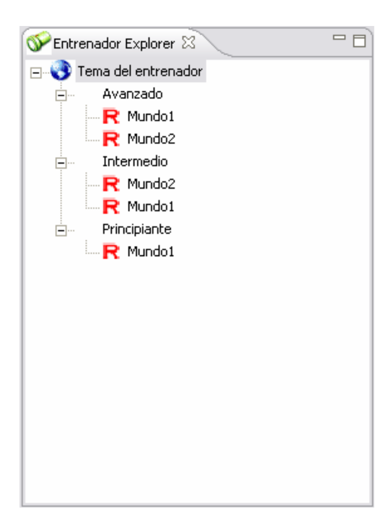
La Perspectiva MicroMundo, esta compuesta por una serie de vistas que componen al Entrenador Brazo Mecánico. Para acceder a dicha perspectiva, se selecciona “Open a perspective” en el workspace de Eclipse y se selecciona la perspectiva MicroMundo:



La perspectiva MicroMundo esta compuesta por varias vistas, las cuales tienen una función específica dentro del Entrenador.

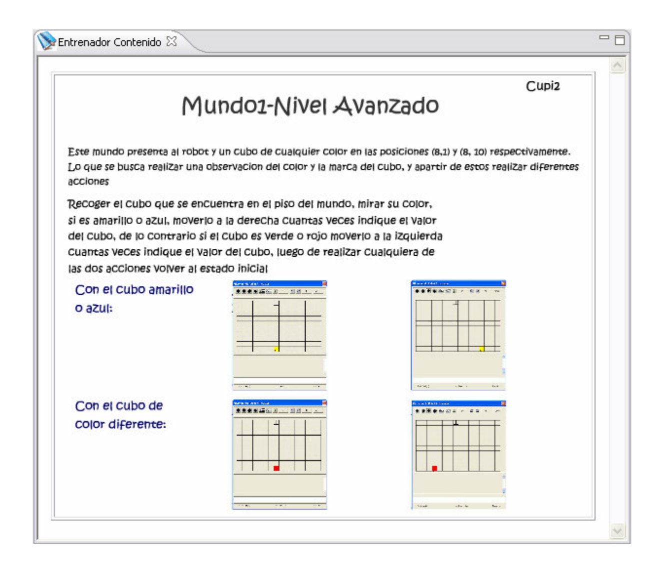


3.1 Vista Entrenador Explorer



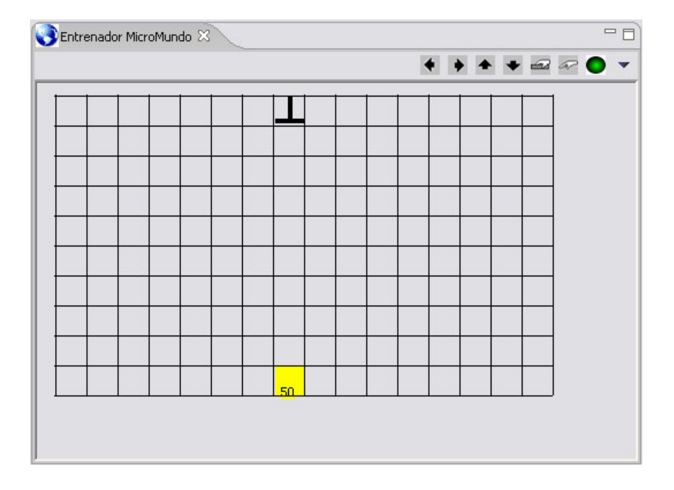
La vista Entrenador Explorer es la encargada de mostrar la información del Entrenador, pertinente a los niveles de dificultad y los mundos contenidos en cada uno de ellos, en esta vista uno puede seleccionar los niveles y los mundos, y puede crear proyectos Java para dar solución a los retos planteados en cada mundo según su dificultad.

3.2 Vista Entrenador Contenido



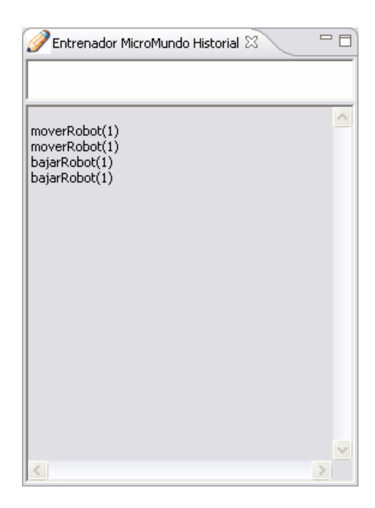
La vista Entrenador Contenido es la encargada de desplegar toda la información pertinente al mundo seleccionado en la vista Entrenado Explorer, en la vista Entrenador Contenido se muestra el reto del mundo, el estado inicial y el estado final que se desea para cumplir el reto propuesto.

3.3 Vista Entrenado MicroMundo



La vista Entrenado MicroMundo es la encargada de visualizar el robot Brazo Mecánico y el mundo en el cual interactúa, en esta vista se puede ver la ejecución de las soluciones dadas a los retos planteados.

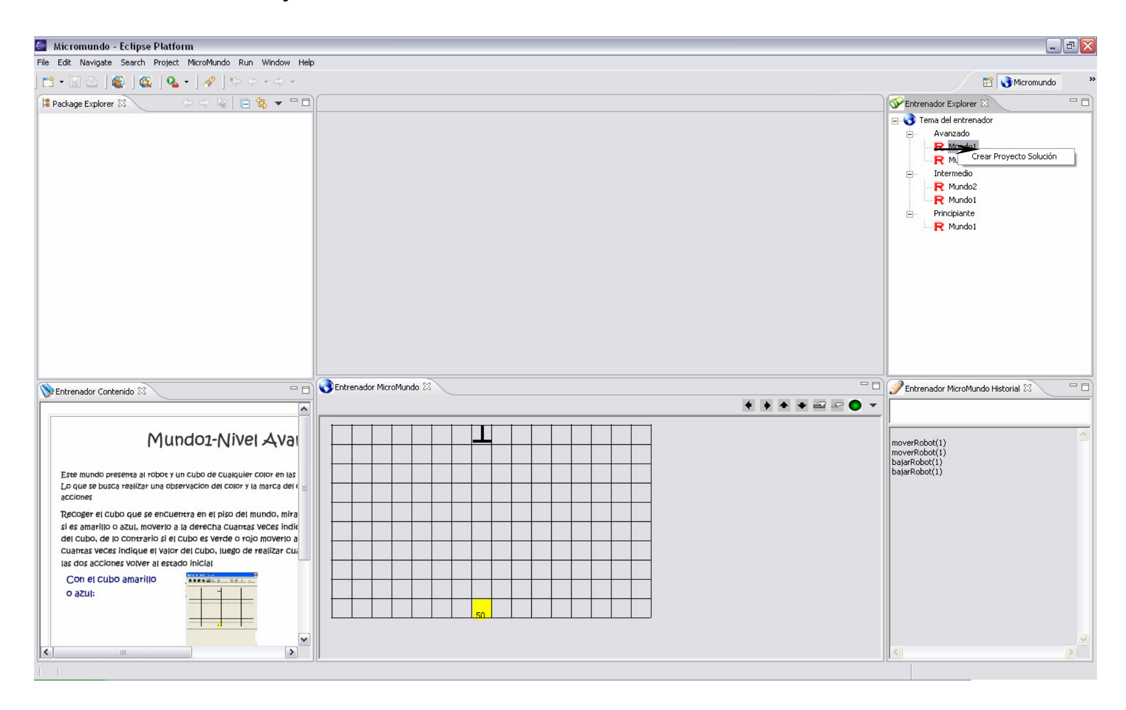
3.4 *Vista Entrenado MicroMundo Historial*



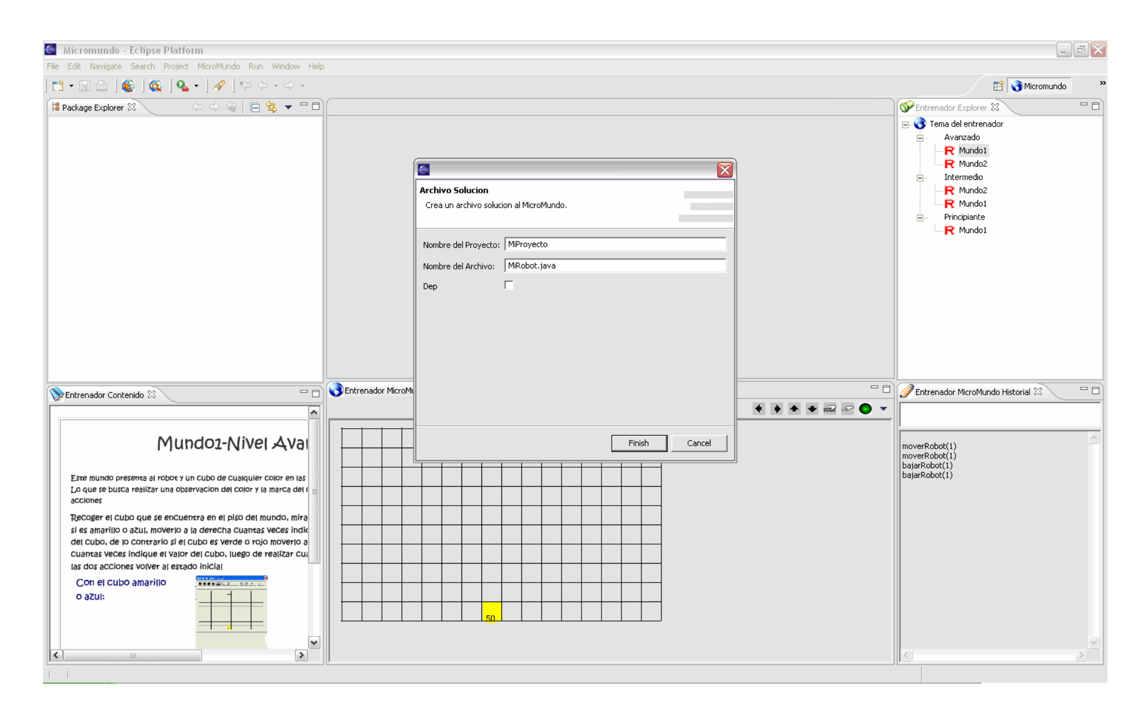
La vista Entrenador MicroMundo Historial es la encargada de mostrar el historial con todos los movimientos realizados por el brazo mecánico, también se pueden ingresar comandos escritos para ser ejecutados por el Brazo Mecánico.

4 Crear un proyecto solución

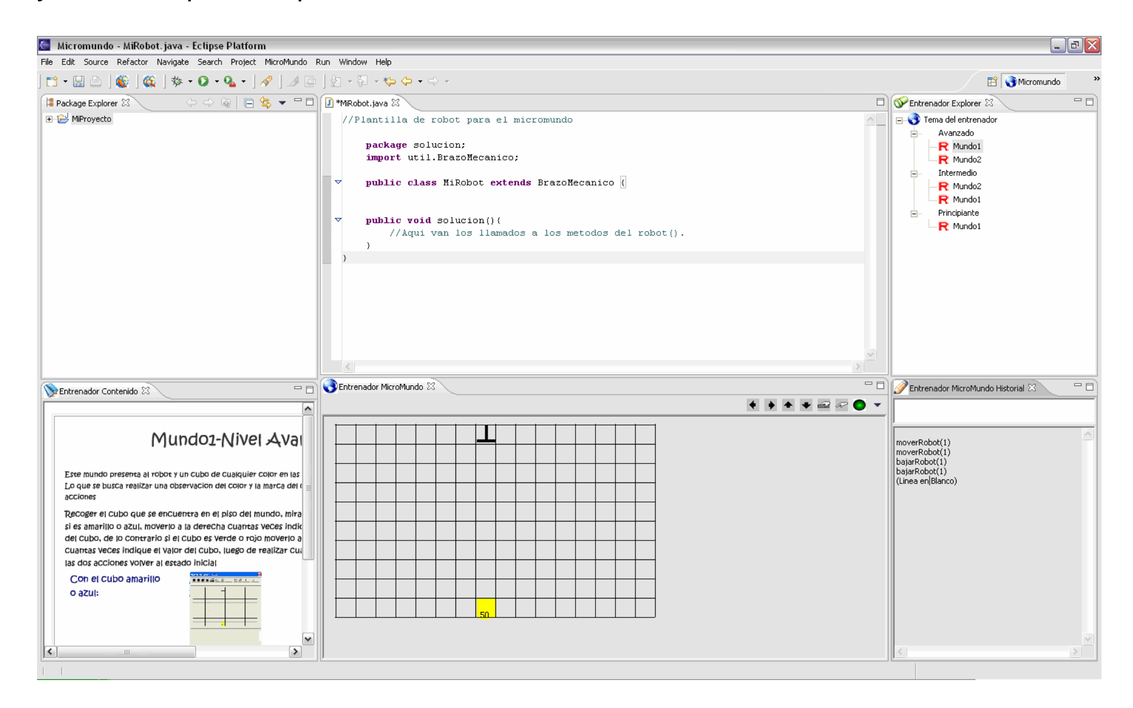
Se selecciona en la vista Entrenado Explorer el mundo al cual se desea crear un proyecto solución, luego se oprime con el botón derecho del Mouse, y se le da clic a Crear Proyecto Solución.



Aparece el Wizard para crear el archivo solución, en este se da el nombre deseado para el proyecto y para el archivo solución

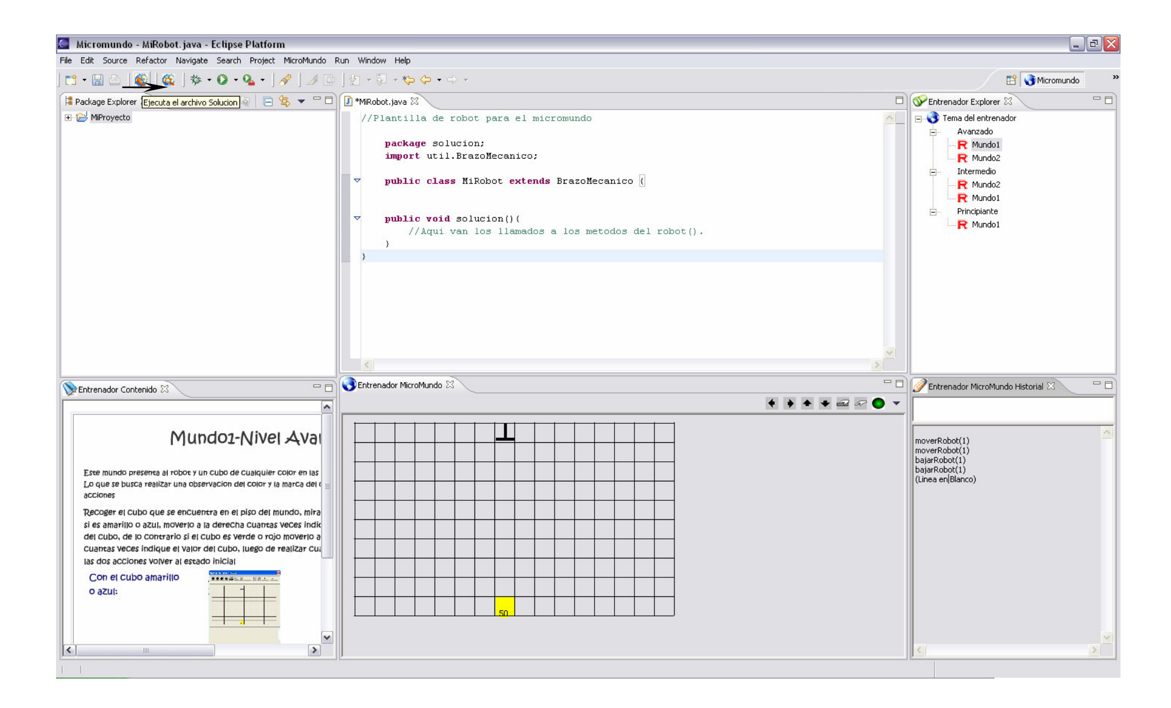


Finalmente de abre el editor y se visualiza el proyecto en el Package Explorer, y esta listo para empezar a escribir la solución al reto seleccionado.



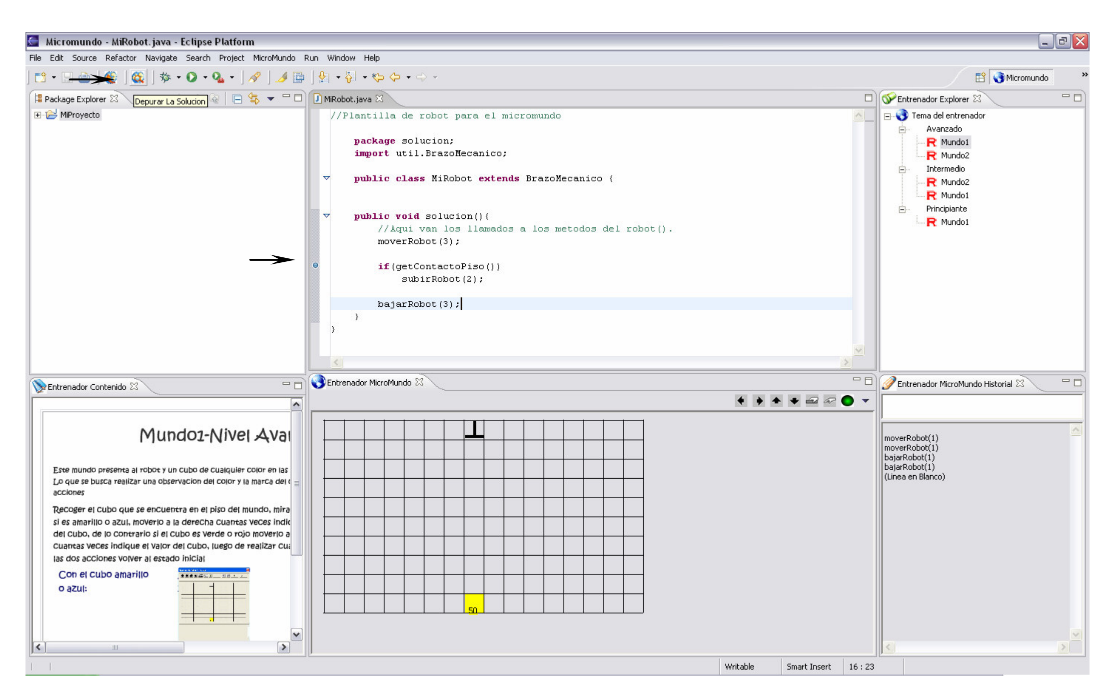
5 Ejecutar un proyecto solución

Cuando se desee ejecutar el archivo solución, se realiza clic en el icono de ejecutar el archivo solución.



6 Depurar un archivo solución

Cuando se desea depurar un archivo solución, se deben insertar los break points deseados para que pare la ejecución normal de la solución, y realizar clic en el icono depurar solución para empezar la ejecución del archivo solución.



7 Comandos Brazo Mecánico

Los comandos para interactuar con el brazo mecánico son los siguientes:

□. public void moverRobot (int pasos)
Mueve el robot horizontalmente un número de pasos determinado @param pasos número de pasos que se desea mover el robot. Si pasos es positivo, el movimiento es a la derecha; si es negativo, es hacia la izquierda

□. subirRobot(int pasos)
Sube el robot el número de pasos determinado.
@param pasos Número de pasos que se desea subir el robot.
Debe ser positivo

□. public void bajarRobot(int pasos)
Baja el robot el número de pasos determinado.
@param pasos Número de pasos que se desea bajar el robot.
Debe ser positivo

□. public boolean llevaCubo()
Devuelve un valor booleano (true o false) que indica si el robot lleva un cubo o no
@return valor que indica si el robot lleva un cubo o no

□. public int getPosicionX()
Retorna la posición en X del Robot

□. public int getPosicionY()
Retorna la posición en Y del Robot

□. public String getColorCubo()
Devuelve el color correspondiente al cubo que lleva el robot.
@return Valor correspondiente al color del cubo. Puede ser "clRed" (rojo),
"clBlue" (azul), "clGreen" (verde) o "clYellow" (amarillo).

□. public int getMarcaCubo()
Retorna la marca del cubo, que es un valor entero mayor a 0

□. public int getDistanciaCubo()
Retorna la distancia vertical que hay del brazo al primer cubo en la misma posición en X del brazo. Si no hay un cubo retorna 1

□. public int getDistanciaCubo()
Retorna la distancia que hay del brazo al cubo en pasos. Si no hay un cubo retorna 1

□. public boolean getContactoHorizontal(String direccion)
Devuelve un valor booleano que indica si en la dirección indicada el robot está en contacto por la parte lateral con un cubo o con el límite del mundo @param direccion Cadena que indica la dirección horizontal en la que se desea verificar el contacto. Debe ser "derecha" o "izquierda"

- public boolean getContactoVertical ()

Devuelve un valor booleano que indica si el robot está en contacto por encima de un cubo

- public boolean getContactoPiso()

Devuelve un valor booleano que indica si el robot está en contacto con el piso (tocando el piso)

- public void tomarCubo()

Hace que el robot tome el cubo que está a su alcance, es decir exactamente debajo de él.

- public void soltarCubo()

Hace que el robot suelte el cubo que lleva y lo deja en la posición en la que se encuentra. No se puede soltar un cubo en una posición mayor a 1.